Fermi National Accelerator Laboratory

FN-312
2530.000

A MODULAR TRIGGER PROCESSING SYSTEM FOR HIGH ENERGY PHYSICS EXPERIMENTS

E. Barsotti, J. Appel, M. Haldeman, R. Hance, B. Haynes,
J. Maenpaa, T. Nash, T. Soszynski and K. Treptow
Fermi National Accelerator Laboratory, Batavia, Illinois 60510

S. Bracker
University of Toronto, Toronto, Ontario

October 1978

Operated by Universities Research Association Inc. under contract with the Energy Research and Development Administration

A Modular Trigger Processing System For High Energy Physics Experiments

E. Barsotti, J. Appel, S. Bracker*, M. Haldeman
R. Hance, B. Haynes, J. Maenpaa,
T. Nash, T. Soszynski, K. Treptow

Fermi National Accelerator Laboratory†
Batavia, Illinois 60510

*University of Toronto
Toronto, Ontario

Section 1: Introduction

We will describe in this paper the design of a powerful new high level trigger processing system that is flexible enough to be used in a wide variety of high energy physics applications. The system is based on a newly evolving technique of using large fast access memories as table look-ups in place of combinational logic. It uses high speed emitter coupled logic (ECL) and memories and is capable of making trigger decisions requiring complicated calculations in a few microseconds. It is totally modular in concept representing an extension in power and flexibility of NIM fast logic philosophy rather than a fast reduced capability version of a small computer. This system has no computer like program counters, clocks, instruction registers, etc. which unnecessarily slow down fast trigger decision logic. Programming the processor is done by cabling modules together at the front panels, as in NIM fast logic, and by loading control information into modules via CAMAC. The modules operate in an unclocked, non handshake mode. Far more powerful than fast logic, these modules operate on multi bit words, finding tracks, calibrating pulse heights and computing complicated functions. Single module operations typically take 50 nanoseconds. Modular RAMs (Random Access Memories) allow table look-up of arbitrary functions. They are also used to control program flow, aborting processing when certain conditions have been met, for example. These RAMs may be loaded and checked through CAMAC. Simple logic functions (ANDs and ORs) are handled with General Logic Modules.

All internal data is transmitted using differential ECL drivers and receivers over single and multiconductor twisted pair cables. Hard wire fanouts are not permitted. However, many modules provide buffered outputs of input data allowing fanout via daisy-chaining of modules.

Although the system operates in a high speed unclocked mode, system algorithms may include hardware processing of nested loops, conditional branching and subroutines, operations normally associated with serial computer-type processing. Unclocked operation means that each module only waits for data specifically required at its inputs; thus processing is completed in the shortest possible time. Operation is controlled by a system of Ready lines. When all required input Readys to a given module are set, the module starts its operation. When a module has completed its operation, the module sets its output Ready line. In almost all applications, inter module timing need not be a concern of the user. The difficulties of timing between two asynchronous loops, with one loop feeding data to the other, are handled automatically by the Stack module functioning as a FIFO or a randomly readable buffer. The user need only interconnect modules to set up the desired algorithm and load memories with appropriate functions and tables.

To permit system testing, nearly every intermodule data path can be monitored or controlled by CAMAC read and write commands. The system is packaged using CAMAC hardware and a modified CAMAC crate with an ECL backplane and a CAMAC TTL to ECL translator in slot 23. As seen from the host computer the system responds and transmits according to CAMAC standards.

A total of eighteen different modules have been designed. These include ten general purpose modules of varying complexity such as Memory Look-Up, Data Stack, General Logic, Fan Out and level conversion modules. There are four test modules, three limited purpose modules associated with track finding, and one special purpose module relevant only to the initial experimental application of the system. Many of these modules will be described in the following sections. Further details and full documentation on the modules, hardware, etc., may be found in ECL/CAMAC Trigger Processor System Documentation, Fermilab Technical Memo TM-821.

Multi-Level Triggers

A multi-level triggering philosophy is fundamental to the concept of powerful modern trigger processing. Conventional NIM logic is used to make a loose "low-level" trigger on a time scale of ~100 nsec. This trigger is used to gate data recording electronics such as CAMAC ADCs, TDCs, latches, etc. If the low level trigger rate is kept below 10^4/sec, for example, then high level decisions on whether or not the data should be recorded by the computer may take as long as 10μsec resulting in only 10% dead time. This is long enough for very sophisticated computations, using the fast integrated circuitry and memories now available, to reduce the data recording rate to the order of 100 events/sec. If the high level trigger decision is negative a clear signal is sent to the CAMAC data recording electronics. Most modern commercial CAMAC modules are capable of a fast clear in 1-2μsec. The trigger processing system described in this paper is designed to make high level trigger decisions of this type. In its initial application, to be described below, the processor will trigger experiments in the Tagged Photon Beam at Fermilab. At maximum rates in this beam there will be no more than 6000 hadron producing photon interactions per second. The background, some 2000 times larger, is electron pair production. Experience with the measurement of the total hadronic photon cross section allows a straight-forward low level NIM trigger of all hadronic interactions with little pair contamination at ≤ 6000 triggers/second. This trigger is described elsewhere in discussions of the photon total cross section experiment.[1] The trigger processor, in the worst case of this application, thus has 16μsec (for 10% dead time) to select a subset of interesting events to be recorded on tape from all the hadronic photon interactions triggered by the NIM logic.

An Application Example: The Tagged Photon Spectrometer

The processor has been designed as part of the Tagged Photon Spectrometer now under construction at Fermilab.[2] The forward part of the spectrometer consists of two large magnets, drift chambers, segmented Cerenkov counters, a large area segmented shower

†Operated by the Universities Research Association, Inc under contract with the United States Department of Energy.

counter, a hadron calorimeter and μ identifiers. The initial application of the processor will be triggers based on the recoil system. We will use this application as an example showing the broad flexibility of the processor system. The recoil system is shown in cross sectional views in Figure 1. It consists of three concentric, two-meter long, cylindrical multiwire proportional chambers (MWPC) surrounding a hydrogen

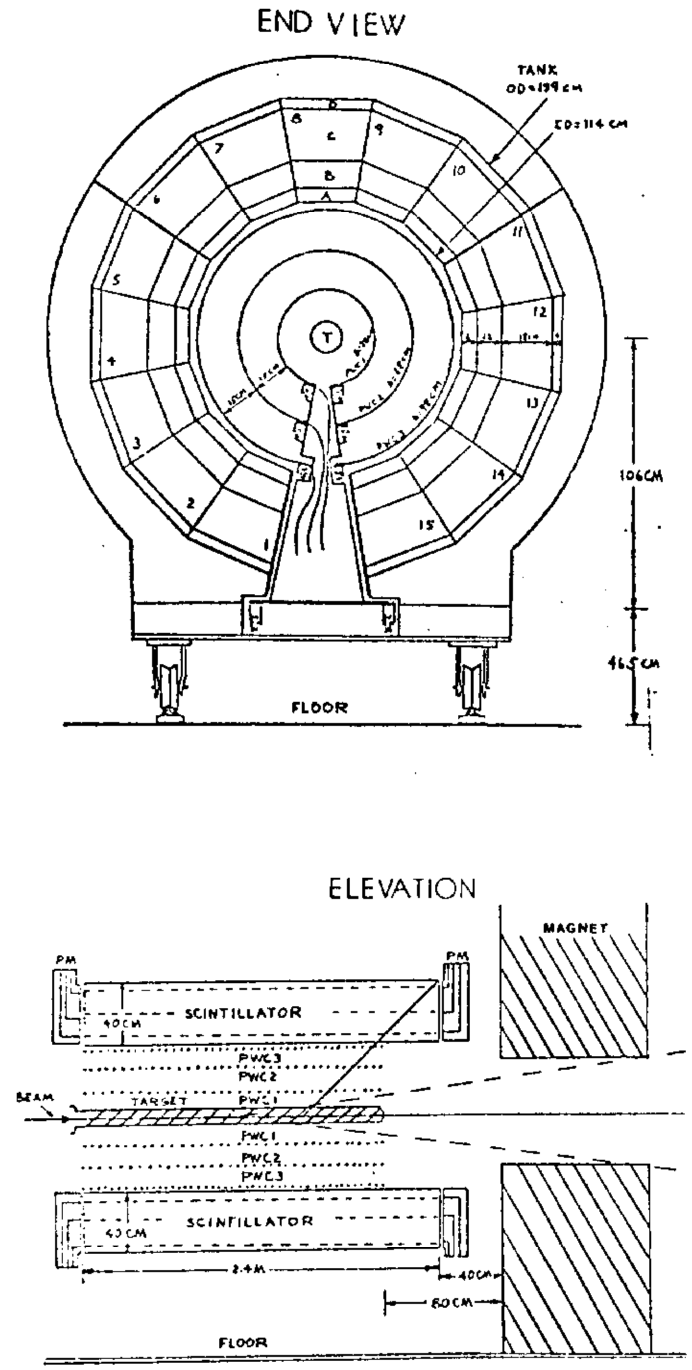

Figure 1
Recoil System

target (which may be up to 2 m long). To simplify track computations, the target to inner chamber distance is equal to the inter chamber spacing. Surrounding the MWPC are scintillation counters in 15 azimuthal sections and four radial layers. The inner two layers are plastic and the outer two are liquid. In the MWPC the anodes run parallel to the target, spaced every few millimeters azimuthally. The outer cathodes of the MWPC consist of mylar on which a grid of wires has been wound and glued at 3mm spacing. The mylar is bent and glued to the inside of a honey comb support cylinder. Thus the wires, which run in circles perpendicular to the beam, measure Z the distance along the beam line. The signal induced by charge flowing toward the anode is read out on these cathode wires. The wire chambers provide a means for measuring the polar angle, Θ, of tracks recoiling from interactions in the target. The energy deposited by particles traversing the scintillators is also measured. Recoiling protons of typical angle with kinetic energy between about 50 and 250 MeV stop in the scintillator and thus their total energy, E, is readily measured. With somewhat more difficulty the energy of protons over 250 MeV can be determined from the $\frac{dE}{dX}$ information measured in the scintillators. This information can also be used to identify the recoiling particle as a proton. Even if nothing is measured directly in the forward going system, x, in $\gamma p \to xp$, the missing mass, M_x, can be computed from the measurements of the recoil proton E and Θ if the photon energy E_γ is known. E_γ is measured by the photon tagging system. system[1].

The trigger processor system will be used first to select events with only a proton in the recoil detector. This strongly enhances the sample of diffractively produced forward forward going states. Then the processor computes the missing mass and triggers on a selected range, for example to enhance diffractive charm events, $4 < M_x < 5$GeV. The system is capable of handling a reasonable number of spurious tracks (δ rays) and secondary interactions. It is estimated that an average of 10μs will be required for this trigger.

Briefly the recoil trigger system will operate as follows (refer to Figure 2). Data from the MWPC is transmitted to the processor. The Track Finder subsystem scans all MWPC hits searching for those that form a straight line, called a track, in the three chambers. When a track is found, its slope ($\alpha = \cot\Theta$) and intercept with the target center line, the vertex (V_z), are computed and stored. As soon as tracks are available another loop of the processor associates each track with one of the 15 scintillator sectors. This is done by computing the projected Z coordinate of the intersection of the track with the inner scintillator ring and comparing with end to end timing measurements of hits in the inner scintillators. The timing measurement gives the Z position of tracks in the scintillator to ± 1 inch. If a match is found, the sector number is determined. At this point, digitized pulse heights for all scintillators have already been stored.[3] The sector number is used to select the four scintillator pulse heights for the sector that the track passed through. The pulse heights are calibrated into units of energy. This information is the energy deposited in each scintillator and is examined first to check for consistency with proton energy loss patterns and then to determine the track energy. There is now enough information to compute the missing mass for this track. However, all relevant tracks are examined. The track from the most upstream vertex is stored. If there is one and only one proton track from the most upstream vertex the processor computes M_x and generates one or more trigger signals if M_x falls in one of the predetermined ranges.

We now turn to a detailed description of the Trigger Processing System. As the various modules and

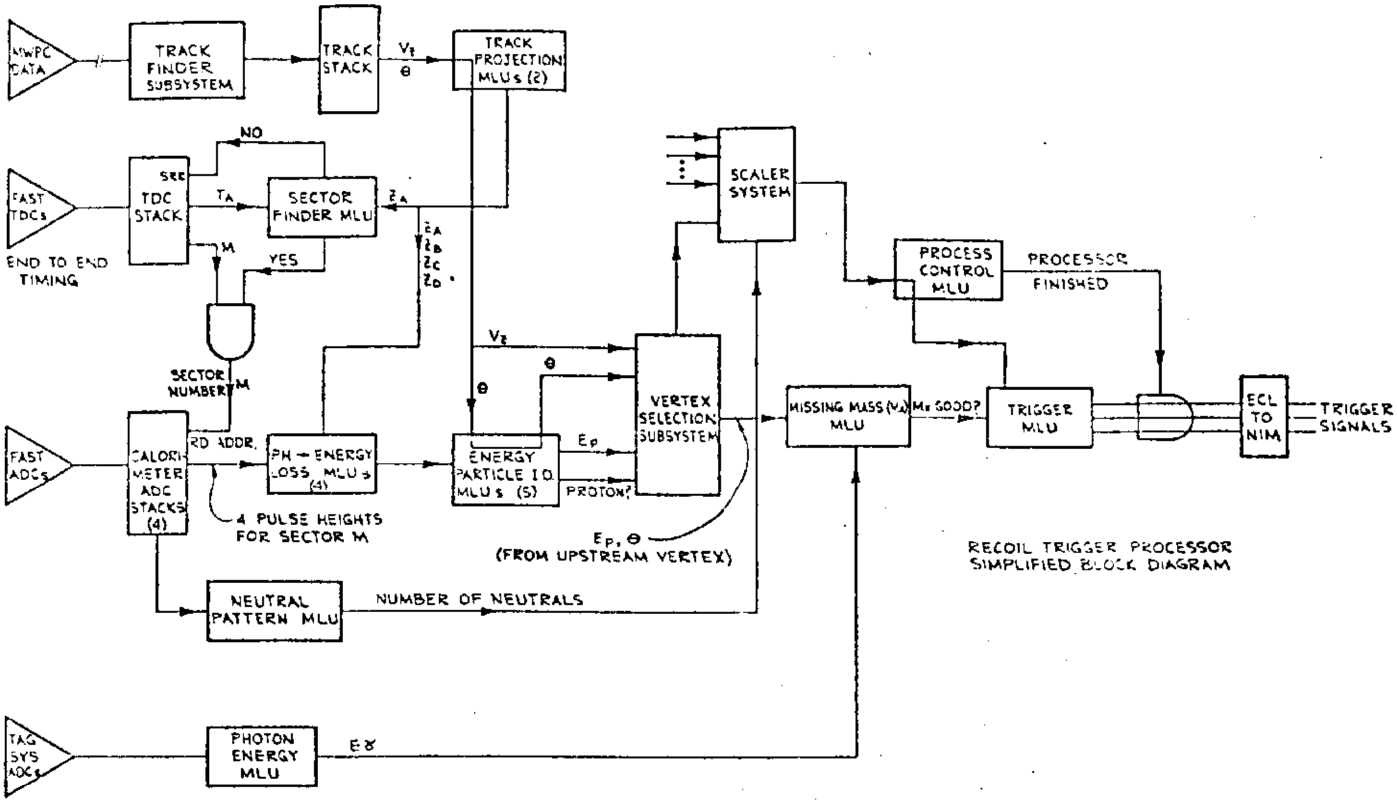

Figure 2

subsystems are described, details of the recoil trigger will be explained as examples.

Section 2: Packaging

The problems associated with packaging of electronic circuitry are quite often grossly underestimated. The foremost design objectives of the trigger processor were that it be fast and that the design be general in nature so that the hardware will be both easily adaptable to the dynamic requirements of ongoing experiments and reusable and mass-producible for future experiments. These goals dictated a modular system implemented with high-speed ECL technology. In addition, data usually is read from different sources and several modules require two way communication with the host computer. Because of the processor's size and complexity, it was necessary to provide easy mechanisms for both on-line and off-line diagnostics. These additional requirements and the general availability of CAMAC hardware and software led to the packaging scheme shown in Figure 3. Standard CAMAC parallel Branch Highway hardware is used along with Fermilab-Standard software CAMAC I/O routines.

ECL CAMAC Crate and Power Supply

The ECL CAMAC crates differ from standard CAMAC crates only in the backplane and the method for connecting the crate to a power supply. The standard TTL crate backplane has been replaced by a single-ended 100Ω ECL backplane. Only the low order sixteen read/write data bits are used. The TTL to ECL translation is done by a module in slot 23 thus allowing standard Type A CAMAC crate controllers to be used. The translator module additionally functions as a Dataway Display module for local diagnostics. If the CAMAC backplane had not been changed to ECL, many additional integrated circuits, over twenty in some cases, would have been required on each module to interface to the CAMAC system and translate from ECL to open-collector TTL signals.

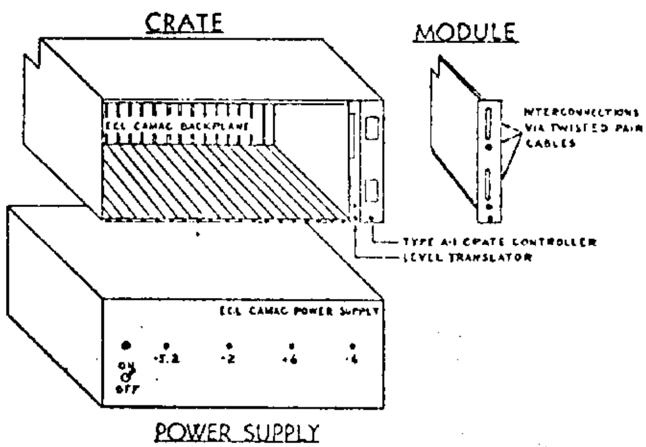

Figure 3

The ECL CAMAC power supply consists of four commercial supplies packaged in a nineteen-inch, rack-mountable case seven inches high. The voltage outputs and current capabilities are listed in Table 1.

Voltage	Maximum Output Current 40°C	60°C	Regulation
-5.2V	100A	65A	Switching
-2.0V	50A	35A	Switching
+6.0V	9A	7.5A	Linear
-6.0V	2.5A	2A	Linear

TABLE 1

The power supplies are all overvoltage and overcurrent protected; the -5.2V and -2.0V supplies are remote sensed at the ECL crate backplane.

The crate and power supply are cooled by forced air from external fans. Maximum allowed power dissipation per double-width module is 50 watts and per crate is 500 watts. New modules should not dissipate more than 30 watts per single-width of front panel space.

ECL CAMAC Module Construction Techniques

Owing to the high-speed properties of ECL-10000 circuitry, all system modules were constructed using a microstrip technique, i.e., wire over a ground plane, to provide nearly uniform signal transmission characteristics within the modules. The three module construction techniques used in the project are described and compared below.

An ECL wire-wrap CAMAC kludge-board was designed for the project. It has space for up to 62 sixteen pin ICs of which up to eight may be replaced with 24 pin ICs. Most modules needed in low quantities (including diagnostic modules) were wire-wrapped using this board. The characteristic impedance of signal lines on these wire-wrapped boards varied from 100Ω to 140Ω; 120Ω terminating resistors were used.

Modules required in larger quantities and designed in the project's early stages were constructed using three-layer printed circuit (PC) boards with the middle layer a ground plane. Power distribution for these modules is by mini-busses above the board. The characteristic impedance of signal lines on these boards holds fairly constant at 100Ω (i.e., two oz. copper, ten mil signal line width and 1/32 inch from signal line to center ground plane). This method of construction has two disadvantages. First, due to high-speed requirements, signal paths should not contain forks which add long stubs to the transmission lines. Signals must flow from a driver to the first receiver, then to the next receiver, and so on to the terminating resistor. This requirement limits the number of ICs on a given PC board area to roughly 60% of the number on a conventional TTL design. The second disadvantage is that unless the circuitry is minimal, short stubs cannot be avoided when taping an ECL PC board. Thus speed is unavoidably degraded.

A third and relatively new construction method, suitable for low to medium quantity board requirements, was used. This method uses a two-sided power and ground plane PC board each side covered with a thin material adhesive on its outer side. Circuitry is wired point to point by a machine laying small insulated wire in the adhesive. The material thickness determines the spacing between the circuitry and signal ground and thus the characteristic impedance. This technique is called "multi-wire" and provides the best environment for high-speed ECL circuitry. There are no transmission line stubs and the characteristic impedance of signal lines again is fairly constant. Multi-wire boards generally have a nominal impedance of 56Ω but this can be different if so desired. Table 2 compares the relative parameters of the three construction techniques.

Construction Technique	Wire-Wrap	Three-Layer PC	Multi-Wire
Operating Speed (time)	1.0	0.7	0.6
IC Density	1.0	0.5	0.8
Cost(smaller quantities)	1.0	2.0	1.6

TABLE 2

Module Interconnections and Asynchronous Timing Requirements

The ease of interconnecting modules along with the ability to program modules via CAMAC helps meet the design objective of flexibility. Flexibility is necessary so that the processor system will be adaptable to future needs and to the dynamic requirements of ongoing experiments. All interconnections between trigger processor modules and almost all interconnections between these modules and other experimental electronics is via ECL differential drivers and receivers. All interconnections are made at the modules' front panels using 110Ω multiple twisted-pair ribbon cable with industry standard flat ribbon connections for data and 110Ω single twisted-pair cable with LEMO connectors for input Ready strobes.

Since the system is totally asynchronous (i.e., not sequential) one need only connect the desired combination of data signals and their accompanying input Ready signals to the input connectors. Critical timing problems are thus non-existent. It takes only a few minutes to assemble the necessary specially configured ribbon data cables, using mass termination techniques. Thus, much like the NIM standard used heavily in high-energy physics experiments today, quick physical rearranging of modules and data inputs and outputs is possible with this hardware.

Section 3: The Modules

As stated previously, an important objective of the trigger processor project was to design the modules, as much as possible, to be fast but general in nature so that they can be both reused and mass-produced for other experiments. The following subsections will give a brief overview of all the modules, detailing a few modules and showing that this objective was met.

Module Summary

Table 3 summarizes the modules designed and built for the first application of this trigger processor system.

Of the eighteen different types and 108 modules built, sixteen types and 102 modules are general in nature and can be used in other experiments, eliminating the major portion of project design and development time.

All modules were specified and designed so that they could be computer diagnosed either individually or as part of subsystems or as a complete system. Additionally all subsystems and the system as a whole can be single-stepped by the host computer providing an additional level of diagnostic capability. These testing mechanisms make it easy to uncover failures both inside modules and in connecting cables. Routine monitoring of system operation during experimental running can be handled by the host computer. Monitoring and diagnostic software are discussed further in a later section.

Several of the system's important modules as well as the track finder subsystem will now be described.

Memory Look-Up Module - The Heart of the Trigger Processor

The Memory Look-Up module (MLU) is a high-speed table look-up device. The availability of high density fast, low cost solid state random access memories

Module Name	Quantity	Approximate Parts Cost	Width	Construction Technique	Remarks
TTL/ECL Level Translator	10	$ 275	1	Three-layer PC	General system module
Memory Look-Up	25	1200	2	Three-layer PC	General system module; up to 16 4096-bit RAMs.
ECL CAMAC Test Module	4	150	2	Wire-wrap	General test module.
Stack	12	500	2	Multi-wire	General system module, 32 16-bit words.
Active Extender	2	50	1	Wire-wrap	General test module.
PWC Data Receiver/ Centroid Processor	1	400	2	Wire-wrap	General system module.
PWC Test Transmitter	2	300	2	Wire-wrap	Specific test module.
Do Loop Indexer	2	275	2	Wire-wrap	General system module; processes as data is received.
Track Finder	4	1050	2X2	Wire-wrap	Somewhat specific but easily applicable to different geometries; two modules.
Do Loop Controller/ Test Transmitter	3	150	2	Wire-wrap	General system and test module.
Quad Scaler	8	525	2	Multi-wire	General system module; one to four bit inputs.
Vertex Parameter Module	2	275	2	Wire-wrap	Specific system module.
General Logic Module	16	225	1	Three-layer PC	General system module; equivalent of several NIM modules.
Fan Out Module	8	75	1	Two-sided PC	General system module.
TDC Data Receiver	2	100	2	Wire-wrap	General system module.
Input Level Converter	2	200	1	Commercial CAMAC	General system module NIM to ECL converter.
Output Level Converter/ Trigger Module	2	100	2	Wire-wrap & Two-sided PC	General system module, ECL to NIM converter.
Signal Monitor	2	100	2	Wire-wrap	General test module.

TABLE 3

(RAMs) makes it possible to precalculate functions of arbitrary complexity, store results in tabular form, and, when needed, access the appropriate location in memory to obtain a real time answer in 50 nsec or less. At this writing the cost of 35 nsec, 4096 bit ECL RAMs is down to one cent per bit. In high energy physics and other applications more and more complicated high speed functions will be handled with programmable memory chips rather than with fixed hard wired logic circuits.

The MLU is designed to bring the ability to use modern high speed memory to the high energy physics user in a flexible and easy to use package. It is an extremely general device, whose function can be transformed completely through downloading from the computer. It accepts $\leq$ 16 bits of input which is used to address the internal memory. The number in the addressed word of memory is the MLU output. With the MLU the designer obtains the little box often portrayed in beginning calculus classes - the general function machine into which one inserts a vector within the legal domain and obtains a vector result which may be arbitrarily defined within the legal range. We are still learning how to use this generality most effectively. For instance, we have found it useful at many points to characterize data fields using unusual scaling - neither linear nor logarithmic. A two bit field may have four possible states mean 'less than 2', '2 through 10', '11 through 30', 'very big'. Note that both input and output data fields may include several variables allowing multi-dimensional functions.

Each MLU contains either 16384 bits (using 16 1K RAMs) or 65536 bits (using 16 4K RAMs). This memory may be organized in five different ways; the organization is selected by a front-panel switch on the MLU. The output words may be 1, 2, 4, 8 or 16 bits wide. As the output word length increases, there are correspondingly fewer words provided and consequently fewer address (MLU input) bits. Table 4 summarizes the options:

Output Word Width	Input Address Width		Maximum Address	
16	12	(10)	4096	(1024)
8	13	(11)	8192	(2048)
4	14	(12)	16384	(4096)
2	15	(13)	32768	(8192)
1	16	(14)	65536	(16384)

TABLE 4

Quantities in parentheses refer to MLUs with 1K RAMs as memory.

Each MLU has several input Ready lines. Each module contributing to the MLU input indicates that its portion is valid by driving an input Ready line true. (Unconnected input Ready lines are considered true). When all inputs are valid, output data is generated and a short time later output Ready is signaled. The time from the last input Ready to output Ready is typically 35 ns for modules having 1K RAMs and 50 ns for modules with 4K RAMs. Removing any input Ready drops output Ready, but the data stays valid until input data is changed.

Although the output word length is selectable, loading and reading an MLU through CAMAC is always done using 16-bit words. Software is being written to load MLUs which will make it unnecessary for the physics user to know the internal memory organization. The user need only specify which bits in the input and output ports are used for various data, and write a Fortran subroutine which returns the output vector given an input vector; the program will deliver a table of numbers suitable for loading directly into the MLU. Other routines load the numbers from the table into the MLU and read back the numbers as a check.

In concluding this section we list examples of MLU applications that demonstrate its versatility.

Fast time or pulse height calibration.
High speed arithmetic (+, -, x, ÷).
Process control.
Conditional branching.
Trigonometric and logarithmic function.
Single module, programmable, timing independent replacement of conventional experimental NIM logic.

Later we will discuss specific applications of the MLU in the Recoil Trigger for the Fermilab Tagged Photon Spectrometer described earlier.

The Stack

The stack was designed to store blocks of data which could be sequentially or randomly accessed asynchronously with write cycles. The data written into the stack is often raw data coming from various detectors such as multi-wire proportional chambers and scintillation counters whose outputs have been converted to digital signals. The Stack may also be used as a FIFO buffer between asynchronous loops. In such cases the data stored is output from other modules such as the MLU, Track Finder or anywhere fast access data storage is required.

Each module is capable of storing thirty-two, sixteen bit words. Several modules may be used together to expand the storage in either the word or bit direction.

Data is written into the Stack by presenting it at the Write Data (WD) input simultaneously with a data valid signal at the Sequential Write Ready (SWR) input. Each SWR strobes in data and increments the write address pointer in preparation for the next write.

Data can be read from the Stack either sequentially or randomly. A random read is executed by supplying a five bit address to the Read Address input simultanesouly with a signal at the Random Read Ready input. This action causes the Output Ready signal to disappear followed by the appearance of the requested data at the Read Data output and the reappearance of the Output Ready signal indicating valid data.

A sequential read is initiated by a signal at the Sequential Read Ready input. The output response is similar to the Random Read except that the read address comes from an internal counter which is incremented when the Output Ready goes high.

Since the internal RAM is a single access memory, logic in the module sorts out the write requests and the two types of read requests and stores them until they can be handled. The stack alternates between reads and writes when both are present by setting priority for the next request based on the type of request currently being serviced. For fast uninterrupted block writes to the stack, a read inhibit line is provided which locks out random or sequential reads.

It is expected that production versions of this module will be able to store data in intervals of about 25 nsec. The intervals for reading are expected to be about 33 nsec.

Do Loop Indexer

This module acts as a controller that cycles through all values of two indices (I, K) much like a Fortran DO loop except that the upper limits may increase while the loop is in operation. Each pair of I > 0, K > 0 (or KMIN(I) as explained below) up to the upper limits at the end of data transmission is presented on the I, K outputs once and only once. Upper limits at any moment are flagged externally by signals at the I Too High (ITH) or K Too High (KTH) inputs to this module. A K Too Low (KTL) input is also available to set the K lower limit (KMIN(I)) for all higher I. KTL is used, for example, to avoid wasting time on backward tracks as discussed in the next section. The module is able to control the random read of data from two stacks (for track finding, for example) while the stacks are still being filled with data (from MWPC readout, for example). A description of the use of this module in the Track Finding Subsystem may be found in the following section.

At the beginning of each cycle the module examines the input control bits (ITH, KTH, CONT, KTL, etc.). Using backing registers it quickly outputs new indices. Simplified, the algorithm followed is this:

1. If neither ITH or KTH is true but CONT is received, then the new I is equal to the old I and K is incremented.

2. If KTH is true but ITH is not, I is incremented and K is set back. For an I that has never been tried before, K is set to zero; for an that has already been partially examined, K is set to the value of K that last caused KTH to be set. No I, K pair will be transmitted twice

3. If ITH is true, I is set back to zero and K is set back to the last value of K tried on the previous pass at this I. If all data was in the last time I was set back, then the pass thru all the indices just performed is the last required, and ITH will cause a DONE signal No more indices will be sent.

The Track Finder Subsystem

The Track Finder Subsystem was designed to find straight line track segments from particles traveling through three evenly spaced wire chambers. These may be planar or concentric as in the recoil system described earlier and shown in Figure 1. The three chambers measure the coordinates of the points where the chambers are crossed by a particle trajectory, Z_I^{in}, Z_J^{mid}, and Z_K^{out}, respectively. The system can also be used without change to correlate hits in U, V, X wire chamber systems, where U and V refer to coordinate axes typically $\pm 20^\circ$ from X. The Track Finder Subsystem is shown in Figure 4. In the recoil system several wires fire at the site of each hit. The readout circuitry for each chamber provides a list of hits, characterized by the number of adjacent wires in each cluster and the position of the center of the cluster. The Track Finder accepts this as data and outputs a list of tracks found, characterized by the slope of the track (cot Θ) and position of its intersection with the beam line, the assumed interaction vertex (V_Z).

The centroid address and widths of groups of wires which have been hit during the current event are

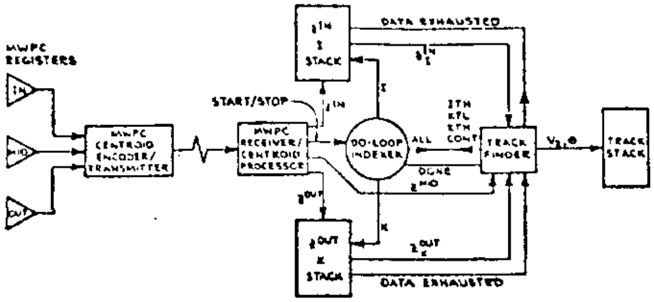

Figure 4
Track Finder Subsystem

transmitted at rates up to one hit per 80 nsec to a PWC data receiver/centroid processor module. Details of the PWC data transmitter designed at Carleton University by L. Bird and W. Ferguson will be described elsewhere. The PWC data receiver processes the centroid/width data, sorts it according to which chamber it came from, and creates hit information for the trigger processor. The PWC data receiver sends the I and K hits addresses to stack modules where they are recorded in ascending order. Hits from the middle chamber (J) are sent to a hit array in the Track Finder module. The Do Loop Indexer module selects indices I and K and presents them as Random Read Addresses to two Stacks containing Z_I^{in} and Z_K^{out} which are then presented to the Track Finder Module. The latter module then calculates

$$PZ^{mid} = (Z_I^{in} + Z_K^{out})/2$$

which is the projected location in the middle chamber if this I, K corresponds to a track. As middle chamber (Z_J^{mid}) data is received it is used as a 10-bit address to a 4 X 768 bit RAM called the hit array ($H(Z_J^{mid})$). This location is loaded with $H(Z_J^{mid}) = I_{ev}$, a 4-bit number (never 0000) that increments on each event and acts as an event identifier. After the projected mid-chamber position (PZ^{mid}) is calculated it is used to address the hit array. If $H(PZ^{mid}) = I_{ev}$ (for the present event), a track is found and the following quantities which have been simultaneously computed are presented to two stacks which act as a FIFO buffer to the rest of the system:

$$\alpha = Z_K^{out} - Z_I^{in} \quad \text{(the track slope)}$$

$$S\alpha = \begin{cases} 1 & \alpha \geq 0 \\ 0 & \alpha < 0 \end{cases}$$

and

$$V_Z = \frac{3Z_I^{in} - Z_K^{out}}{2} + OFF$$

V_Z is the Z coordinate of the track intercept with the beam at one chamber spacing unit inside the inner chamber. OFF is an alignment constant settable via onboard switches which must be set so $V \geq 0$ for good tracks. These quantities are computed simultaneously with PZ^{mid} and are available as soon as the track decision has been made. If V<0 the track is vetoed. For each J centroid received the PWC Data Receiver module can be programmed to load more than one coordinate into the hit array. This provides the ability to adjust the track finding criterion to correspond to wire chamber measurement resolution. This module is also capable of being programmed to treat MWPC centroids with large widths (several wires on) as more than one hit and to add an alignment constant to the coordinates of J hits.

Data is being received by $H(Z_J^{mid})$ and the Z_I^{in}, Z_K^{out} stacks simultaneously with the track processing operations described above. The chamber data must be loaded into the stacks and hit array in an ordered way (upstream first, downstream last, for example). If P_Z^{mid} is greater than the most recent (and largest) Z_J^{mid} received by the hit array, KTH (K Too High) is set to 1. If the Do Loop Indexer has attempted to index either stack beyond data received the stack will raise its Data Exhausted (DE) level. This is received by the Track Finder module and ITH (I Too High) or KTH will be raised, as appropriate. These signals are used by the Do Loop Indexer to determine the next indices. In addition if a switch selectable option is chosen, KTL (K Too Low) will be set and tracks ignored if $\alpha < \alpha_{min}$, a parameter which may be set via on board switches. This allows quick rejection of uninteresting tracks. If a track is found or, if not found, and none of the KTH, KTL, ITH conditions exist, then CONT (continue) is set to instruct the Do Loop Indexer to proceed.

The reason the hit array is 4 bits wide (instead of 1) is that large fast RAMs have no general clear. Thus to clear the hit array completely would take 768 loads. Even at 15 nsec each this would take over 10μsec and cause an unacceptable dead time. Instead roughly 1/15 of the array is cleared after each event. Since there are 15 event identifiers (I_{ev}) all locations of the array with I_{ev} from the last time a given I_{ev} was used will have been cleared before the same I_{ev} is used again.

The time taken by the Track Finder Subsystem loop (which includes index generation, Z^{in} and Z^{out} stack access time and the Track Finder Module processes time) is approximately 110 - 120 nsec if the indexed data is available from the stacks and 70 nsec if not.

There are two aspects of the Track Finder design which contribute to its relatively high speed operation. First is the ability to start processing MWPC hits before all the hit data has been received. The Do Loop Indexer is capable of keeping up with continuously changing upper limits as the data comes in. Second is the use of the hit array and looping over only two indices rather than three. If there are as many as 10 hits per chamber and a typical cycle takes approximately 100 nsec, the processor takes 10μsec using two indices, 100μsec using three.

Quad 4-Bit Scaler

This module is intended for scaling of internal operations and loops as well as the frequency of occurance of various internal conditions. Examples of the latter in the recoil trigger are failed tracks, non-proton tracks, and the number of different vertices. The results may also be transmitted via CAMAC on each event to the on-line computer to monitor changes in such things as success ratios that would indicate a hardware problem. The modules are normally reset to zero at the beginning of each event.

In the primary application of this module as a scaler, each of the four sections performs scaling (or adding) of up to 4 bits of input data. The output of each section is 4 bits. An overflow bit, one per section, is set via a jumper selectable option when an output equals or exceeds a count of 1, 2, 4, 8, or 16. The user may select, via another switch option, to freeze the output and overflow of a given section at the occurance of an overflow condition or to use the overflow bit to cascade the sections providing up to a 16 bit counter within one module. The scaler can count as fast as 40 MHz.

Each section uses an ECL 10181 IC arithmetic logic unit (ALU) to scale input data. By using this ALU and on board switches this module is additionally capable of performing several arithmetic and logic functions such as incrementing, decrementing and complementing a single input word. It can also be used for adding, subtracting, normal and exclusive ORing/NORing, ANDing and NANDing of two four bit input words.

The two input registers in each section may be read and written via CAMAC. The output of each ALU is also CAMAC readable. Each section has individual input Readys and Clears. There is a General Module Clear and a General Input Ready. This general Ready may be ORed to any input Ready via on board switches.

General Logic Module

This module is similar to several commercial NIM logic modules but uses ECL levels and is significantly less expensive. It is used as a workhorse unit where isolated gates are required. It contains three sections of logic each capable of the following four logic functions, selectable using on board switches:

$$(A_i \cdot B_i \cdot C_i \cdot G_i \cdot EN_i) + (ON_i \cdot \overline{EN}_i) = OUT_i$$

$$((A_i + B_i) \cdot C_i \cdot G_i \cdot EN_i) + (ON_i \cdot \overline{EN}_i) = OUT_i$$

$$(((A_i \cdot B_i) + C_i) \cdot G_i \cdot EN_i) + (ON_i \cdot \overline{EN}_i) = OUT_i$$

$$((A_i + B_i + C_i) \cdot G_i \cdot EN_i) + (ON_i \cdot \overline{EN}_i) = OUT_i$$

where i = 1, 2, 3, and;
- EN_i is CAMAC controllable and is used to enable (or disable) the front panel inputs.
- ON_i is CAMAC controllable and is used to control the state of OUT_i when the front panel inputs are disabled.

A_i, B_i, C_i, G_i are front panel input signals. Each unused G_i input is interpreted as being in an on state, while unused A_i, B_i, C_i inputs are assumed to be in an off or zero logic state. CAMAC readable switches are used to select logic functions in each section.

Note that the module can generate a number of logic functions and, via CAMAC, can be used to single-step through any subsystem for diagnostics. This is accomplished by inserting one section of logic between a Ready signal input and the Ready signal itself.

Section 4: Diagnostics, Monitoring and Host Computer Software for the Trigger Processor

Hardware, whether at the module or subsystem level, must be developed in association with the preparation of software used in its checkout and later in its operation. In this section we describe our software and the knowledge we have gained of the relationship of software to hardware development and system operation.

Checkout software has been prepared for all the modules we have designed. This will make future production of these designs easier. Operational software, loading MLUs from a PDP-11 for example, will also be available.

A physics user of this system will be concerned with two software areas. One is the programming of MLUs whose memories contain the functions or tables that control the physics of the trigger. A general purpose MLU loading software package is being prepared. This will call subroutines prepared by the user for each MLU. Examples of MLU programming in the recoil trigger are described at the end of this section.

Secondly, the experimental user must be convinced that the system as configured does what is intended and may wish to prepare a computer simulation of any new triggering system. A simulation is probably only necessary for more complex systems. It could be used occasionally during running conditions to compare with processor results. Continuous monitoring of processor performance is probably best handled by using Quad Scaler modules to count various internal processor conditions (such as the number of false tracks) and reading these through CAMAC into the experimental on-line computer on every event. The experimenters and on-line software, can monitor these scalers (as is done for other detector parameters) looking for sudden changes or unusual conditions that indicate equipment failure.

The system being described is a new approach to experimental triggering involving a significant jump in complexity in an area where careful scientific practice is essential. Because of this, we have chosen to bias toward overkill in our diagnostic and monitoring capabilities. In the following we describe the four broad areas of software development for the processor.

Simulation and Design Verification Studies

Simulation and design verification studies were required in studying the operation of the Track Finder, particularly the Do Loop Indexer. Although the function served by the Indexer is quite simple to explain, the internal timing required to obtain very high speed is tricky.

The state diagram of the Do Loop Indexer was first embodied in a Fortran program. This program was run against thousands of randomly generated events. The output convinced us that the basic organization of gates and registers was (or in a few cases was not) sound and the output was complete enough that it was generally not too hard to understand the reason that certain attempts at minimizing the design failed. At least in the case of this module, it would have been a serious mistake to proceed without the simulator. As hardware was produced, the simulator slowly turned into a diagnostic program which calculated what was to be done and then exercised the hardware to insure that the hardware did what was expected. Eventually, all of the modules in the Track Finder were incorporated into the simulation, so that everything from the PWC receivers through the Track Stack could be exercised as a unit.

Simulation of a different kind is required to verify the loading of the various Memory Look Up modules. This is not a simulation of hardware performance as such, but rather a means of determining that physics is being done correctly by the numbers placed in the MLUs. As explained later, the numbers to be placed in the MLUs are first placed in disk files. The simulator runs can predict the output of a chain of MLUs by generating input data and then following the path, referring to the disk files to get MLU contents as required. System level hardware checkout will be incorporated into this simulator as well.

Low Level Diagnostics

Low level diagnostics, performed on a single module or small groups of modules have been designed at two levels. During development, an interactive control language was used to make simple tests and verify basic functions. Alternating patterns of data are written to and read from memory. Such a facility is absolutely essential in first-level checking. At this stage in testing, it is extremely difficult to write a general program to exercise the module in every conceivably useful way. However, when a module has passed these simple tests, it is then necessary to apply tests at a higher level with a program which exercises the module at high rates, using highly randomized patterns. This brings out problems that af- affect only obscure cases, quirky timing and cross-talk problems and intermittent components. Once a module has passed this point, we can be quite certain that it really works, and it can be incorporated into the system.

Low level tests of the second kind are programmed in Fortran using a common set of CAMAC diagnostic routines. Checking any given module configuration involves generating a single subroutine which actually performs the tests. CAMAC handling, error message generation etc, are all handled in common service routines. About two pages of code will usually suffice to exercise a module thoroughly. It has become very clear in the course of the project that both kinds of low-level diagnostics are essential. Developmental debugging cannot be successful with an inflexible Fortran diagnostic. Modules cannot really be verified without using a high-rate and complex diagnostic.

System Level Diagnostics

High level diagnostics develop naturally out of the simulation studies by incorporating hardware checking into the simulation. These are meaningful only after the hardware has already passed the low level tests. We do not see many module hardware errors with these programs. They are essential for checking complicated subsystems like the Track Finder and for making sure that the system as a whole is connected properly. (It is probably true that most failures will be cable problems). They are perhaps most helpful in forcing the overall configuration design to be thought through in detail to the point where it can be programmed.

Monitoring of physics data accepted and rejected by the processor provides the ultimate systems level check on the correctness of the inter-module wiring, of what has been loaded into the memories, as well as of hardware failures or errors. Additionally the processor records in the Quad Scalers and in CAMAC readable latches information related to why events are accepted or rejected. As noted earlier this information can be monitored on an event by event basis to detect subtle or drastic changes resulting from hardware failures.

Loading Programmable Modules

The fourth type of software required is that used to load the MLUs and otherwise configure the processor so that it does physics. Most of the procedures for generating the MLU contents are straightforward; several quantities come into the MLU through the address port and several leave through the data port; the physicist writes a subroutine that embodies a particular function. The subroutine is called by the MLU Loading Program which finds out (from a disk file) where the various input and output fields are found within the input or output port, and builds the MLU load accordingly. The subroutine normally is a function of a number of parameters which are stored in a common parameter file. These parameters can be entered from the on-line computer terminal. By using the same parameter file for all MLUs in the system, consistency is assured. An MLU load file is also maintained by the Loading Program. Whenever a parameter is changed or a new subroutine initiated, the subroutine is called and the MLU memory load is prepared and placed in the MLU load file. When appropriate (beginning of the next data run or as requested from the computer console) all MLUs are loaded and verified in a few seconds from the designated MLU load file. At the beginning of each data run the MLU load file, the MLU parameter file, and all other parameters that define completely the processor configuration to be used during the run, are written on the data tape. This provides a permanent record for later analysis. Comments are also recorded to aid in interpreting the processor configuration during experimental analysis. Details on the MLU loading package and its use are provided in the System Documentation referred to earlier.

Programming the MLUs: The Recoil Trigger as an Example

In earlier sections we have described the recoil trigger in general outline and its track finding subsystem in some detail. Once the track parameters are determined much of the burden of the recoil trigger decision falls on the MLU modules. We shall describe here how these units perform the tasks of calibration and correction of scintillator pulse heights, of testing different hypothesis of particle type and kinetic energy, and finally of combining the resulting information and making the trigger decision.

The MLU modules which are responsible for calibration correct for zero signal level (pedestal subtraction), relative gain of the input channels, nonlinear response of the photomultiplier tubes, and the attenuation of signals in the scintillator medium as a function of the track location in the detectors. There is one MLU for each of the four different scintillator layers. The signals from one calorimeter module as well as the position of the track in that module (computed by another MLU) are input to the MLU (seven bits and six bits respectively). The average behavior of the fifteen channels is used to preprogram the memory with all combinations of incoming pulse area and position in the detector. A constant pedestal is subtracted. Next, a simple quadratic correction is made for the saturation of photomultiplier tubes according to the formula:

$$\text{Corrected Signal} = \text{Signal} \times (1.0 + (\sigma \times \text{Signal}))$$

where σ is a saturation coefficient characteristic of the photomultiplier tubes when used in the relevant range of signals.

The attenuation correction is as large as a factor of three or four. Thus, the input seven bits of information, carry only five bits of equivalent information for signals at the farthest part of the detector. For this reason, only five of the available eight output bits are used in this application. (Note that only 10, instead of the maximum 16, RAMs need to be loaded into these MLUs saving $250 per module).

Even much more complicated mapping of input bits to output bits is performed by two other MLU modules in order to identify a particle as a proton with a particular kinetic energy.

The amount of light emitted in scintillation material is proportional to the energy lost and is given by the following transcendental equation (from the Bethe-Block equation):

$$\text{Light} \sim \frac{1}{\beta^2} \{\ell n \frac{\delta}{(1 - \beta^2)} - 2\beta^2 - \gamma\}$$

where β is the particle velocity and δ and γ are constants. A further complication is given by the fact that as the particles transverse the active medium, they loose some of their physically significant original kinetic energy (and, therefore, velocity β). A program was written by G. Hartner to predict the amount of light expected for a particle of given kinetic energy (KE) entering the counter at an angle θ, correcting for energy loss along the way and saturation of light output by the scintillator near zero energy. These results were then fitted to the easily inverted pair of equations given below. For particles stopping in the scintillation counter:

$$\text{Light} \sim \alpha + \exp\{\beta + \gamma \ell n(KE\text{-}KE_0) + \delta \ell n^2(KE\text{-}KE_0)\}$$

For particles penetrating completely through the counter:

$$\text{Light} \sim \alpha' + \exp\{\beta' + X^{\gamma'}\}$$

where

$$X = \delta' - \varepsilon' \ell n(KE\text{-}KE_0) + \phi$$

and all Greek letters are angle dependent parameters. For many values of light intensity, both of the above possibilities (penetrating and non-penetrating) exist. Thus, it is necessary to consider the information from two neighboring concentric layers of scintillator. One MLU receives the corrected energy deposited information for the two inner scintillators and another MLU receives this data for the two outer counters. Both also receive csc Θ. The Fortran subroutines for these MLUs each calculate (for all possible input data to the MLU) four energies. These correspond to the two possibilities (penetrating and non-penetrating) obtained by inverting the two equations above for each of two counters. These energies are fitted to four particle hypothesis, proton, pion, kaon, and electron, and the penetrating vs. non-penetrating possibility. Two bits are set indicating that the track is either possibly, definitely or definitely not a proton. The output also includes the kinetic energy computed for the best hypothesis. The results from the two MLUs (for the inner and outer pairs of counters) are combined and checked for consistency in two other MLUs which output a final track energy and particle type. Note again that the Fortran subroutines for all the MLUs try all input bit combinations and the results are downloaded to the MLU for eventual high speed output during on-line operation.

The energy deposited in the detectors and the resulting digitized information are effectively linear and start from zero. These limitations are not placed on MLU module outputs. The output bits are essentially addresses of output information. Thus, the first output number (bin) might be an overflow or nonsense signal. The next bin might be a few units of kinetic energy and the size of the bins may increase very nonlinearly. In fact, the following bin edge encoding algorithm is used in moving kinetic energy information from one MLU module to another:

$$KE_{out} \sim (\text{Bin Number})^{\text{Power}} + \text{Constant.}$$

This algorithm is matched fairly well to the kinetic energy resolution of the detectors and, therefore, loses no information while compacting a large dynamic range into a few (in this case, five) bits.

Another application of the MLU module is the combination of various trigger information into the final trigger criteria, often with sliding thresholds. In addition to the calculated missing mass observed for a recoiling proton, the existance of other tracks in the recoil system coming from the same vertex, the scaled number of downstream tracks in the recoil system, numbers of neutral particles or low energy electrons observed - all are used as input to an MLU module loaded with predetermined combinations acceptable for triggering. For those events in the missing mass range of highest interest, the other criteria may be set loose. For study, one may trigger with one or more criterion removed and others tightened to get a purer sampling of events. Separate bits corresponding to different types of triggers are used. Some are prescaled external to the processor before being recombined for final master trigger generation.

In addition to the attempted calculations from all possible input bit combinations, the inputs are tested for consistency. In many applications, some input combinations are not physically possible or meaningful. Typically, one number among the possible output numbers (notice, one number rather than one bit) is reserved to signify such data status. At other times, a bit is reserved for this purpose and the processor cycling is aborted rather than follow such configurations to the end.

These examples show how the MLU modules are appropriate for use whenever all possibilities for input information can be preprocessed to give output information. The amount of memory used here is already as large as is found in many on-line computers. This combined with the greater speed and the ability to use off-line analysis in generating output tables gives these devices more power than one can obtain, for example, by using on-line computers for event selection.

Section 5: Summary

The status to date of the Recoil Triggering System is as follows. All modules are designed and production quantities on all but one design have been built in-house or have been received from vendors. Forty percent of the hardware including the Track Finder Subsystem has been tested. System diagnostic software and on-line data taking software are currently being worked on in parallel.

Acknowledgements

We would like to acknowledge important discussions with Prof. Paolo Franzini at the time the idea to undertake this project was conceived. We would also like to thank J. Bronstein, G. Hartner, G. Luste, J. Martin, R. J. Morrison and all the other E516 experimenters for useful discussions and encouragement Thanks also to B. Conrin, P. Jacobs, R. Kolewe, E. Joannou, and R. Muth for technical support. We acknowledge technical discussions with B. Wadsworth and R. Downing. Finally, we thank Laurie Hanabarger for typing the manuscript, and D. Kline and J. Krebs for drafting support.

References

1. D. O. Caldwell, J. P. Cumalat, A. M. Eisner, A. Lu R. J. Morrison, F. V. Murphy, S. J. Yellin, P. J. David, R. M. Egloff, M. E. B. Franklin, G. J. Luste, J. F. Martin, J. D. Prentice, and T. Nash, Phys. Rev. Lett. 40, 1222. John Perry Cumalat, Ph.D. Dissertation, University of California, Santa Barbara, September 1977 (Unpublished).

2. J. Appel, D. Bartlett, S. Bracker, G. Hartner, G. Kalbfleisch, G. Luste, P. Mantsch, J. Martin, R. Morrison, T. Nash, U. Nauenberg, D. Ritchie, K. Stanfield, and S. Yellin, The Tagged Photon Magnetic Spectrometer: Facility Design Report, Fermilab, May 1, 1977 (unpublished).

3. A system of 8 bit ADCs and TDCs capable of 1-2μsec readout is being developed by C. Kerns at Fermilab for this project. Details will be reported at a later date.